Cole Bardin
CS361
9/30/24

HW1

Question 1:

a)

P	Q	Ins	n	P temp	Q temp	i	j
p1	q1	Init	0	-	-	-	-
p2	q1	p1: i = 1	0	-	-	1	-
p2	q2	q1: j = 1	0	-	-	1	1
p2	q3	q2: temp=n	0	0	-	1	1
p3	q3	p2: temp=n	0	0	0	1	1
p1	q3	p3: n=temp+1	1	0	0	1	1
p2	q3	P1: i++	1	0	0	2	1
p3	q3	P2: temp=n	1	1	0	2	1
p1	q3	P3: n=temp+1	2	1	0	2	1
X	q3	Repeat last 3 lines 8x	10	9	0	10	1
-	q3	P1: end for lop & return	10	9	0	10	1
-	q1	Q3: n=temp+1	1	9	0	10	1
-	q2	Q1: j++	1	9	0	10	2
-	q3	Q2: temp=n	1	9	1	10	2
-	q1	Q3: n=temp+1	2	9	1	10	2
-	X	Repeat last 3 lines 8x	10	9	9	10	10
-	-	Q1: end for loop & return	10	9	9	10	10

b)

P	Q	Ins	n	P temp	Q temp	i	j
p1	q1	Init	0	-	-	-	-

p2	q1	P1:i = 1	0	-	-	1	-
p2	q2	Q1: j = 1	0	-	-	1	1
p2	q3	Q2: temp=n	0	-	0	1	1
p3	q3	P2: temp=n	0	0	0	1	1
p1	q3	P3: n=temp+1	1	0	0	1	1
p2	q3	P1: i++	1	0	0	2	1
p3	q3	P2: temp=n	1	1	0	2	1
p1	q3	P3: n=temp+1	2	1	0	2	1
X	q3	Repeat last 3 lines 7x	9	8	0	9	1
p1	q1	Q3: n=temp+1	1	8	0	9	1
p2	q1	P1: i++	1	8	0	10	1
p3	q1	P2: temp=n	1	1	0	10	1
p3	q2	Q1: j++	1	1	0	10	2
p3	q3	Q2: temp=n	1	1	1	10	2
p3	q1	Q3: n=temp+1	2	1	1	10	2
p3	X	Repeat last 3 lines 8x	10	1	9	10	10
p3	-	Q1: end for loop & return					
p1	-	P3: n=temp+1	2	1	9	10	10
-	-	P1: end for loop & return	2	1	9	10	10

Question 2:
a)
Assume f(1) = 0

P	Q	Ins	found	i	j
p1	q1	init	-	-	-
p2	q1	P1: found = false	false	0	1

p2	q2	Q1: found = false	false	0	1
p2	q3	Q2: while not found	false	0	1
p3	q3	P2: while not found	false	0	1
p4	q3	P3: i++	false	1	1
p2	q3	P4: found = (f(1)==0)	true	1	1
p2	q4	Q3: j–	true	1	0
p2	q2	Q4: found=(f(0)==0)	false	1	0
p3	q2	P2: While not found	false	1	0
p4	q2	P3: i++	false	2	0
…					

With this order of execution, the value of f that equals zero was found, but the shared variable was then overwritten by the other thread. After this, the values of i and j diverge from the special value. They will never check the correct value again unless there is an overflow. But in the event of an overflow, this same order of execution could happen again. Then, the loops will repeat and not return.

b)
Assume f(1) = 0

P	Q	Ins	found	i	j
p1	q1	init	false	-	-
p1	q2	Q1: while not found	false	-	1
p2	q2	P1: while not found	false	0	1
p3	q2	P2: i++	false	1	1
p1	q2	P3: found = (f(1)==0)	true	1	1
p1	q3	Q2: j–	true	1	0
p1	q1	Q3: found=(f(0)==0)	false	1	0
p2	q1	P1: While not found	false	1	0

p3	q1	P2: i++	false	2	0
…					

This algorithm has a similar exploit to the first one. Both threads enter their loop and thread P finds the value, but then the found variable is overwritten by thread Q. Then both i and j diverge from the special number, only returning in the event of an overflow; which the same execution order could repeat forever.

c)
Assume f(1) = 0

P	Q	Ins	found	turn	i	j
p1	q1	init	false	1	-	-
p1	q2	Q1: while !found	false	1	-	1
p1	q3	Q2: Await turn==2	false	1	0	1
p2	q3	P1: while !found	false	1	0	1
p3	q3	P2: Await turn==1	false	2	0	1
p4	p3	P3: i++	false	2	1	1
p4	q4	Q3: j–	false	1	1	0
p4	q1	Q4: if(f(j)==0)	false	1	1	0
p4	q2	Q1: while !found	false	1	1	0
p5	q2	P4: if(f(i)==0)	false	1	1	0
p1	q2	P5: found=true	true	1	1	0
-	q2	P1: end while loop & return	true	1	1	0
-	q3	Q2: Await turn==2	true	1	1	0
…						

In this case, thread P finds the value and successfully sets the found variable and returns. However, thread Q gets stuck on the await call, waiting for the turn variable to be set to 2 by thread P. But this will never happen because thread P has now stopped execution. If there was a main thread that tried to join these two threads, the join call would never return. Since thread Q never returns, this algorithm would not work.